Enabling Resilience through Introspection and Virtualization

John Paul Walters, Stephen P. Crago, Vijay S. Pai,
Karandeep Singh, Jinwoo Suh, Andrew J. Younge, Kenneth M. Zick
University of Southern California - Information Sciences Institute
Corresponding author: jwalters@isi.edu

***Abstract*—This paper argues that virtualization coupled with an introspective run-time can be used to enhance the fault tolerance of exascale systems while reducing their dependency on checkpointing. By pairing an introspective run-time with virtualization, we propose that many failures can be predicted and other failures detected and mitigated while maintaining concurrency and without resorting to full checkpointing.**

I. Introduction

Traditionally, HPC system architects have relied on checkpoint/restart strategies coupled with sufficiently long mean times to failure (MTTF) to maintain high utilization at relatively low overheads. As supercomputers grew in size, checkpointing evolved to reduce synchronization overhead, checkpoint storage overhead, etc., but fundamentally has not changed. Techniques such as message logging [1], incremental checkpointing [2], local disk checkpoint storage with replication [3], [4], and dedicated checkpointing backplanes [5] have brought us into the petascale era, but on their own will not carry us towards exascale computing as the MTTF begins to approach zero.

In this paper we argue that virtualization combined with a highly parallel introspective run-time system are key enablers for resilient exascale systems. Virtualization offers unprecedented access to detailed performance characteristics, system status, and an infrastructure that decouples system software (e.g. an OS) from the hardware. With the addition of an introspective run-time, fault tolerance can be enforced as a policy of the run-time. This enables the run-time to leverage an array of fault mitigation strategies based on application requirements, including preemptive live migration, algorithm-based fault tolerance, redundancy, etc.

In this paper we specifically target system failures (failures due to components) and soft errors that result in silent data corruption (SDC). Of course, resilience is only one component of future exascale systems, performance is equally important. This is especially true when applications are virtualized, and portions of the application may be moved dynamically over the course of the application's run-time. We address these performance challenges in a second position paper [6].

II. Current State of the Art

As we accept petascale as mainstream, resiliency has recently becoming an increasing challenge. For instance, in [7] it's noted that double bit flips occur in 1 of the 100,000 DIMMS in a Cray XT5 at a rate of 1-2 times per day, leading to a node's instant reboot and loss of all non-checkpointed data. Hard disks and cooling fans fail much more frequently. As we move to exascale and target orders of magnitude larger resources, these problems (and many others) will likely be exacerbated. Other work has targeted understanding checkpointing at scale, and mitigating its effects [3], [4], [8]–[10]. Checkpointing alone, however, is unlikely to scale to the exascale [11].

III. Proposed Solution and Research Approach

In our vision, resilience is achieved through a policy or set of policies enforced by the run-time system. The introspective run-time sits above the virtual machine, and is responsible for both high level resource management (e.g. mapping virtual machines to nodes) as well as global fault prediction, detection, and correction. However it does so with extensive input from virtual machines, applications, and hardware. Applications are responsible for indicating to the run-time their fault-tolerance requirements as well as any fault mitigation techniques the application will employ in parallel with the run-time. Critically, the fault mitigation techniques may vary over time or even by node. Different phases of the application may have different fault tolerance requirements, and heterogeneous resources may have unique resilience requirements.

Implementing our proposed run-time relies on advances in 3 key areas: realtime system modeling, fault prediction, and fault detection/correction. Each of these are described in the sections that follow.

A. Realtime system modeling

Component operating temperature, humidity, load, and age are all well known causes of hardware failure in today's supercomputers [12]. When heat thresholds are reached, machines are powered off. This has the desired effect of reducing heat, but the undesired effect of reducing availability. Core to our vision is the ability to produce realtime models of the full system behavior on a per-node and per VM basis, enabling the run-time to predict both environmental and application-based trends. We propose the use of lightweight CFD-like simulations, enabling the run-time to monitor and predict environmental conditions within nodes, and to correlate those predictions with an application's run-time characteristics (see also the *Mercury* simulator described by Heath et al. [13]).

In our approach we sample local environmental conditions as well as operating load. We also track rack position and location, as well as the data center layout and feed these inputs to a lightweight simulator. Heat transfer and airflow are modeled and trends are predicted based on the node's current and anticipated workload. This allows the run-time to make tradeoffs based on the resiliency policy requested by the application. By leveraging the run-time's scheduling component, virtual machines can be live-migrated to cooler areas of the data center, or cooler areas of the rack. If migration proves too costly, either in terms of performance or resource availability, the run-time can reduce resource allocation to the virtual machine in order to reduce load and bring temperatures back to within a normal operating envelope.

B. Fault Prediction

Fault prediction has been the topic of a great deal of research since the widespread adoption of virtualization [14]. An accurate fault predictor can preemptively migrate computations (e.g. live migrate) to healthy nodes before a fault manifests, and with the help of virtualization, this can be done in a way that is almost totally transparent to the application. The main challenge, however, is accuracy: monitoring CPU temperature, disk SMART status, etc. provides only a partial picture of the overall system health, and only allows very coarse responses (e.g. machine shutdowns).

We propose an integrated approach, leveraging the environmental model described above as well as hardware hooks (e.g. parity errors, error probabilities described below, etc.) to predict node failure based on current and option system states, enabling the run-time to model solutions before their implementation. This would allow the run-time to predict the thermal effect and performance impact of live migrating a VM to a cooler part of the data center.

More powerfully, by combining the system model with hardware inputs, future hardware failures can be predicted based both on historical data (e.g. earlier application runs) prior to adversely impacting the application. Parity errors can be monitored for likely imminent failure, and individual cores can report error conditions. Because the run-time is architecture-aware, it is also capable of aggregating failure preconditions from heterogeneous components.

We propose a reclassification of hardware from either "good" or "bad" to states of marginal behavior. This is especially critical for silent data corruption. Every computation produced by the hardware should have an associated reliability estimate that can be used or sampled by the run-time to predict likely core failures or soft errors. How the run-time reacts to an anticipated failure will depend both on the application policy in place as well as the overall system health.

C. Fault Detection and correction

Past efforts, particularly in the HPC domain, have assumed a fail-stop model to fault detection, and checkpoint/rollback for fault correction. This is typically the case with today's MPI applications where a single fault triggers a cascading failure throughout the entire computation, forcing each process to rollback to the current checkpoint. At the exascale, we must extend our fault models to include not only fail-stop conditions, but performance, marginal hardware faults, and silent data corruption. We anticipate that addressing these issues will necessarily extend through every layer of future exascale systems, including hardware, the application, and management by the run-time.

How hardware reacts to detected failures will depend on the policy in place between the application and run-time. If a computation carries a high probability of failure, it may be re-executed. Alternatively, some applications may not impose strict correctness requirements on the hardware, instead implementing application-based fault tolerance. Others may leverage very low-power cores capable of checking computations, but too slow to produce them. Yet others may leverage otherwise unpowered cores ("dark silicon") and transition computation to the new core and away from the suspect core.

Regardless of the policy in place, applications cannot be allowed to push the system to its operating extremes for long. Virtualization will enable the run-time to seamlessly disable cores and transition computation without application knowledge. Similarly, the run-time can remove resources (e.g. CPU cycles) from the VM if the application is exceeding tolerable thresholds, or live-migrate the VM.

We also anticipate that applications and application run-times will evolve through the exascale era. MPI-3 is expected to include run-through stabilization for traditional fail-stop errors. For those who desire, this will enable programmers to implement fault tolerance at the application-level. Other programming models will exhibit inherent fault tolerance, for example "big data" MapReduce computations.

Still others will fall back to checkpoint/rollback. We anticipate that the advances described above will help to reduce checkpointing frequency. Further, by leveraging preemptive checkpointing [15], and local storage techniques [3], [4] overhead will reduce further.

IV. Assessment

- Challenges addressed: Our approach addresses the challenge of keeping an exascale resilience without sacrificing performance in the context of unreliable components.
- Maturity: Our approach has not been implemented for high-performance computing, but many of the component technologies, such as virtualization, are maturing quickly, and others, such as introspective run-time systems, have been prototyped in research environments.
- Uniqueness: Leveraging policy-based introspection and virtualization allows a paradigm shift compared to existing HPC resilience strategies.
- Novelty: The combination of virtualization and introspection for exascale systems is novel and has not been implemented or proposed for exascale systems to our knowledge.
- Applicability: Our approach is widely applicable to DOE workloads, and ss virtualization becomes more pervasive, this approach could be leveraged at the petascale to reduce the overhead of fault tolerance.
- Effort: Initial efforts should focus on lowering the virtualization overhead barrier, constructing the system models for fault prediction and faults (to include soft errors) at scale. In the long term, efforts should be made to enable run-time integration of hardware statistics and system models.

REFERENCES

[1] P. Lemarinier, A. Bouteiller, T. Herault, G. Krawezik, and F. Cappello, "Improved message logging versus improved coordinated checkpointing for fault tolerant mpi," in *Cluster Computing, 2004 IEEE International Conference on*, sept. 2004, pp. 115 – 124.

[2] S. Agarwal, R. Garg, M. S. Gupta, and J. E. Moreira, "Adaptive incremental checkpointing for massively parallel systems," in *Proceedings of the 18th annual international conference on Supercomputing*, ser. ICS '04. New York, NY, USA: ACM, 2004, pp. 277–286. [Online]. Available: http://doi.acm.org/10.1145/1006209.1006248

[3] J. P. Walters and V. Chaudhary, "Replication-based fault tolerance for mpi applications," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 7, pp. 997 –1010, july 2009.

[4] A. Moody, G. Bronevetsky, K. Mohror, and B. De Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*. IEEE, 2010, pp. 1–11.

[5] R. Gupta, P. Beckman, B.-H. Park, E. Lusk, P. Hargrove, A. Geist, D. Panda, A. Lumsdaine, and J. Dongarra, "Cifts: A coordinated infrastructure for fault-tolerant systems," in *Parallel Processing, 2009. ICPP '09. International Conference on*, sept. 2009, pp. 237 –245.

[6] V. S. Pai, S. P. Crago, D.-I. Kang, M. Kang, K. Singh, J. Suh, J. P. Walters, and A. J. Younge, "Virtualized cloud computing for exascale performance," Jul. 2012.

[7] D. Fiala, "Detection and correction of silent data corruption for large-scale high-performance computing," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, may 2011, pp. 2069 –2072.

[8] K. Ferreira, R. Riesen, R. Oldfield, J. Stearley, J. Laros, K. Pedretti, T. Kordenbrock, and R. Brightwell, "Increasing fault resiliency in a message-passing environment," *Sandia National Laboratories, Tech. Rep. SAND2009-6753*, 2009.

[9] R. Gupta, H. G. Naik, and P. H. Beckman, "Understanding checkpointing overheads on massive-scale systems: Analysis of the ibm blue gene/p system," *IJHPCA*, vol. 25, no. 2, pp. 180–192, 2011.

[10] A. Bouteiller, G. Bosilca, and J. Dongarra, "Redesigning the message logging model for high performance," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 16, pp. 2196–2211, 2010.

[11] E. Elnozahy, R. Bianchini, T. El-Ghazawi, A. Fox, F. Godfrey, A. Hoisie, K. McKinley, R. Melhem, J. Plank, P. Ranganathan *et al.*, "System resilience at extreme scale," *Defense Advanced Research Project Agency (DARPA)*, 2007.

[12] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 337–351, 2010.

[13] T. Heath, A. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and freon: temperature emulation and management for server systems," in *ACM SIGARCH Computer Architecture News*, vol. 34, no. 5. ACM, 2006, pp. 106–116.

[14] A. Nagarajan, F. Mueller, C. Engelmann, and S. Scott, "Proactive fault tolerance for hpc with xen virtualization," in *Proceedings of the 21st annual international conference on Supercomputing*. ACM, 2007, pp. 23–32.

[15] F. Cappello, H. Casanova, and Y. Robert, "Preventive migration vs. preventive checkpointing for extreme scale supercomputers," *Parallel Processing Letters*, vol. 21, no. 2, pp. 111–132, 2011.